Access Tutorial 10: Parameter Queries

The last few tutorials have been primarily concerned with interface issues. In the remaining tutorials, the focus shifts to transaction processing.

10.1 Introduction: Dynamic queries using parameters

A **parameter query** is a query in which the criteria for selecting records are determined when the query is executed rather than when the query is designed.

For example, recall the select query shown in Figure 4.6. In this query, the results set is limited to records that satisfy the criterion `DeptCode = "COMM"`. If you wanted a different set of results, you would have to edit the query (e.g., change the criterion to `"CPSC"`) and rerun the query.

However, if a variable (parameter) is used for the criterion, Access will prompt the user for the value of the variable before executing the query. The net result is that parameters can be used to create extremely flexible queries.

When the concepts from this tutorial are combined with action queries (Tutorial 11) and triggers (Tutorial 13), you will have a the skills required to create a simple transaction processing system without writing a line of programming code.

10.2 Learning objectives

- What is a parameter query? How do I create one?
- How do I prompt the user to enter parameter values?
- How do I create a query whose results depend on a value on a form?

Last update: 24-Aug-1997

10.3 Tutorial exercises

10.3.1 Simple parameter queries

- If you do not already have a `qryCourses` query like the one shown in Figure 4.6, create one now and save it under the name `pqryCourses`.
- Replace the literal string in the criteria row (`"COMM"`) with a variable (`[X]`).

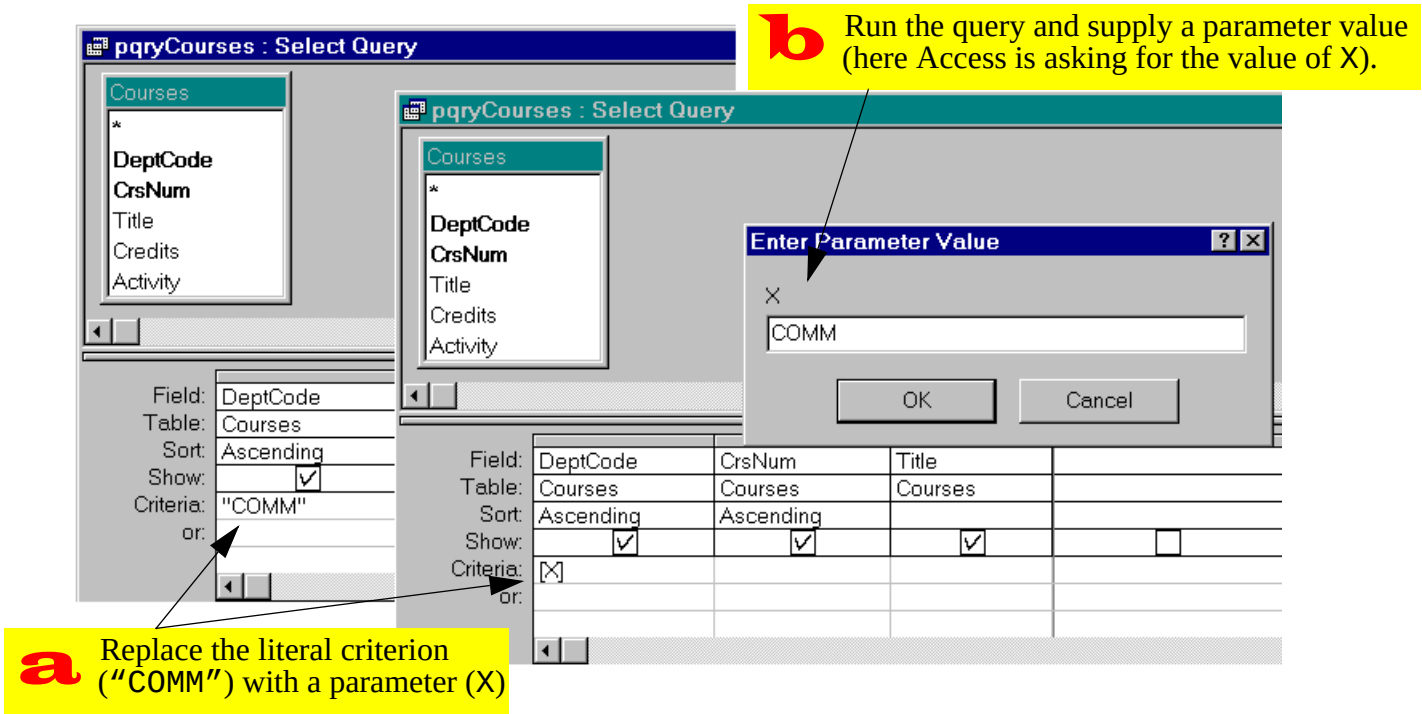
By default, Access expects criteria to be literal strings of text. As a result, it automatically adds quotation marks to text entered in the criteria row. To get around this, place your parameter names inside of square brackets.

- Execute the query as shown in Figure 10.1.

When Access encounters a variable (i.e., something that is not a literal string) during execution, it attempts to bind the variable to some value. To do this, it performs the following tests:

1. First, Access checks whether the variable is the name of a field or a calculated field in the query. If it is, the variable is bound to the current value of the field. For example, if the parameter is named `[DeptCode]`, Access replaces it with the current value of the `DeptCode` field. Since `X` is not the name of a field or a calculated field in this particular query, this test fails.
2. Second, Access attempts to resolve the parameter as a reference to something within the current environment (e.g., the value on an open form). Since there is nothing called `X` in the current environment, this test fails.
3. As a last resort, Access asks the user for the value of the parameter via the "Enter Parameter Value" dialog box.

FIGURE 10.1: Convert a select query into a parameter query.



Note that the spelling mistakes discussed in Section 4.3.4 are processed by Access as parameters.

10.3.2 Using parameters to generate prompts

Since the name of the parameter can be anything (as long as it is enclosed in square brackets), you can exploit this feature to create quick and easy dialog boxes.

- Change the name of your `DeptCode` parameter from `[X]` to `[Courses for which department?]`.
- Run the query, as shown in Figure 10.2.

10.3.3 Values on forms as parameters

A common requirement is to use the value on a form to influence the outcome of a query. For instance, if the user is viewing information about departments, it may be useful to be able to generate a list of courses offered by the department currently being viewed. Although you could use a creatively-named parameter to invoke the "Enter Parameter Value" dialog, this requires the user to type in the value of `DeptCode`.

A more elegant approach is to have Access pull the value of a parameter directly from the open form. This exploits the second step in the operation of a parameter query (Access will attempt to resolve a parameter with the value of an object within the current environment). The basic idea is shown in Figure 10.3.

The key to making this work is to provide a parameter name that correctly references the form object in which you are interested. In order to avoid having to remember the complex naming syntax for objects on forms, you can invoke the expression builder to select the correct name from the hierarchy of database objects.

FIGURE 10.2: Select a parameter name that generates a useful prompt.

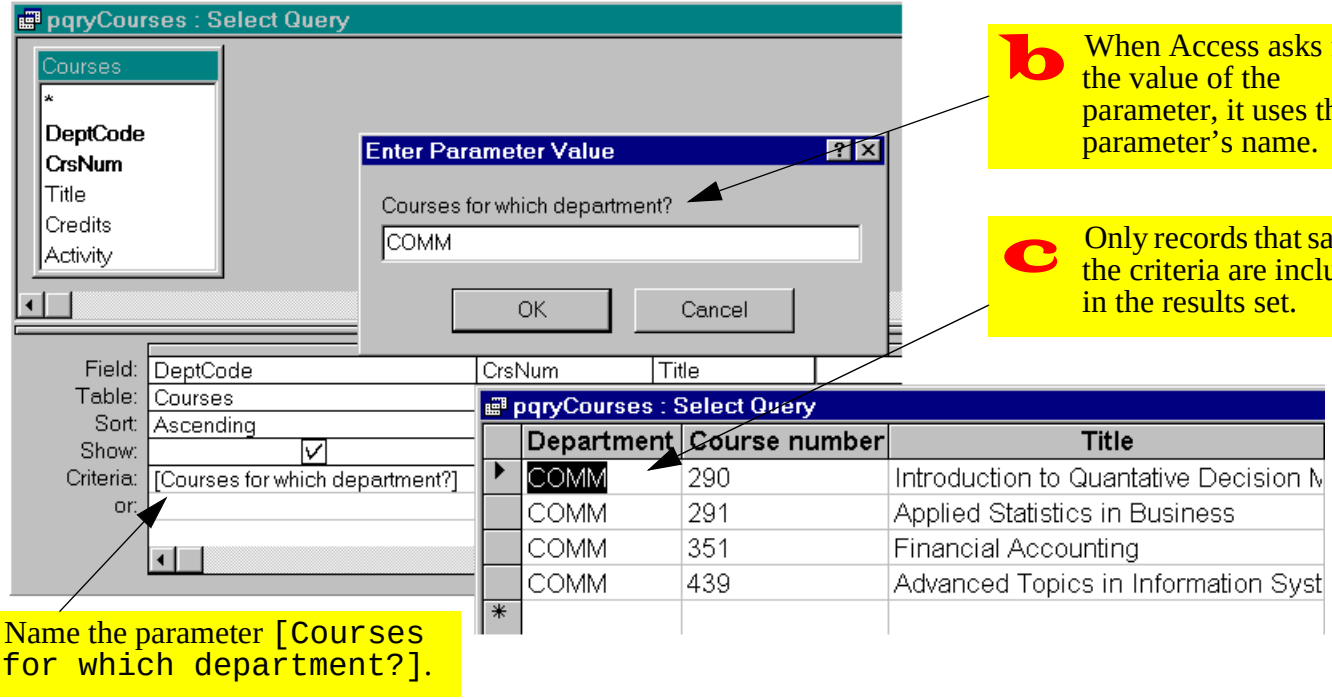
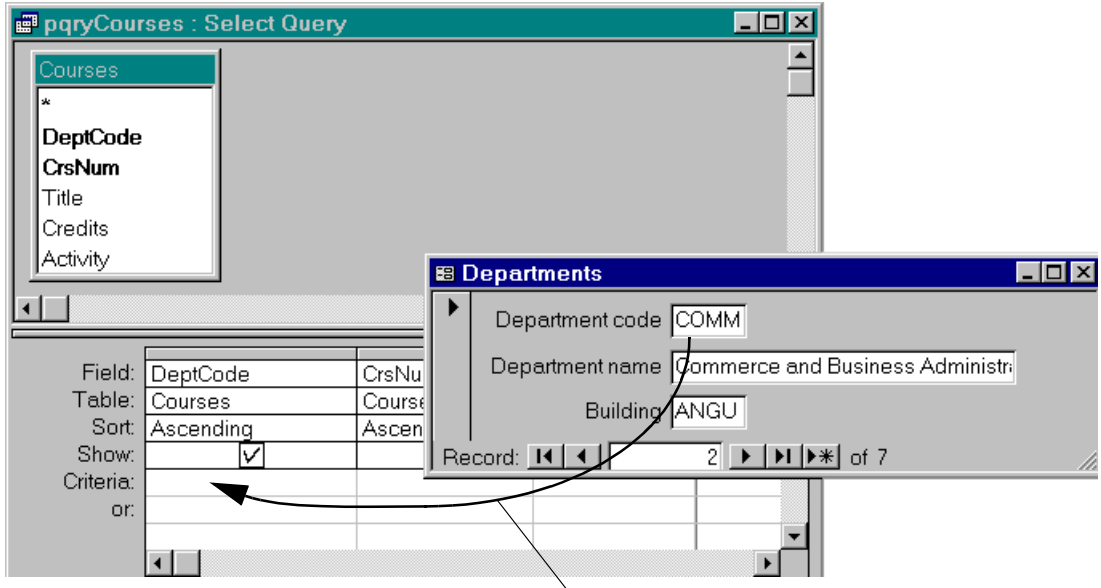


FIGURE 10.3: Using the value on an open form as a parameter in a query.



- Create a very simple form based on the `Departments` table and save it as `frmDepartments`.
- Leave the form open (in form view or design mode, it does not matter).
- Open `pqryCourses` in design mode, place the cursor in the criteria row of the `DeptCode` field, and invoke the expression builder as shown in Figure 10.4.
- Perform the steps shown in Figure 10.5 to create a parameter that references the `DeptCode` field on the `frmDepartments` form.
- Run the query. The results set should correspond to the department showing in the `frmDepartments` form.
- Move to a new record on the form. Notice that you have to requery the form (*Shift-F9*) in order for the new parameter value to be used (see Figure 10.6).

Although the naming syntax of objects in Access is tricky, it is not impossible to comprehend. For example, the name `Forms![frmDepartments]![DeptCode]` consists of the following elements: `Forms` refers to a collection of Form objects; `[frmDepartments]` is a specific instance of a Form object in the Forms collection; `[DeptCode]` is a Control belonging to the form. See Tutorial 14 for more information on the hierarchy of objects used by Access.

10.4 Application to the assignment

You will use parameter queries as the basis for several **action queries** (see Tutorial 11) that process transactions against master tables. For now, simply create the parameter queries that take their criteria values from forms you have already created.

FIGURE 10.4: Invoke the builder to build a parameter.

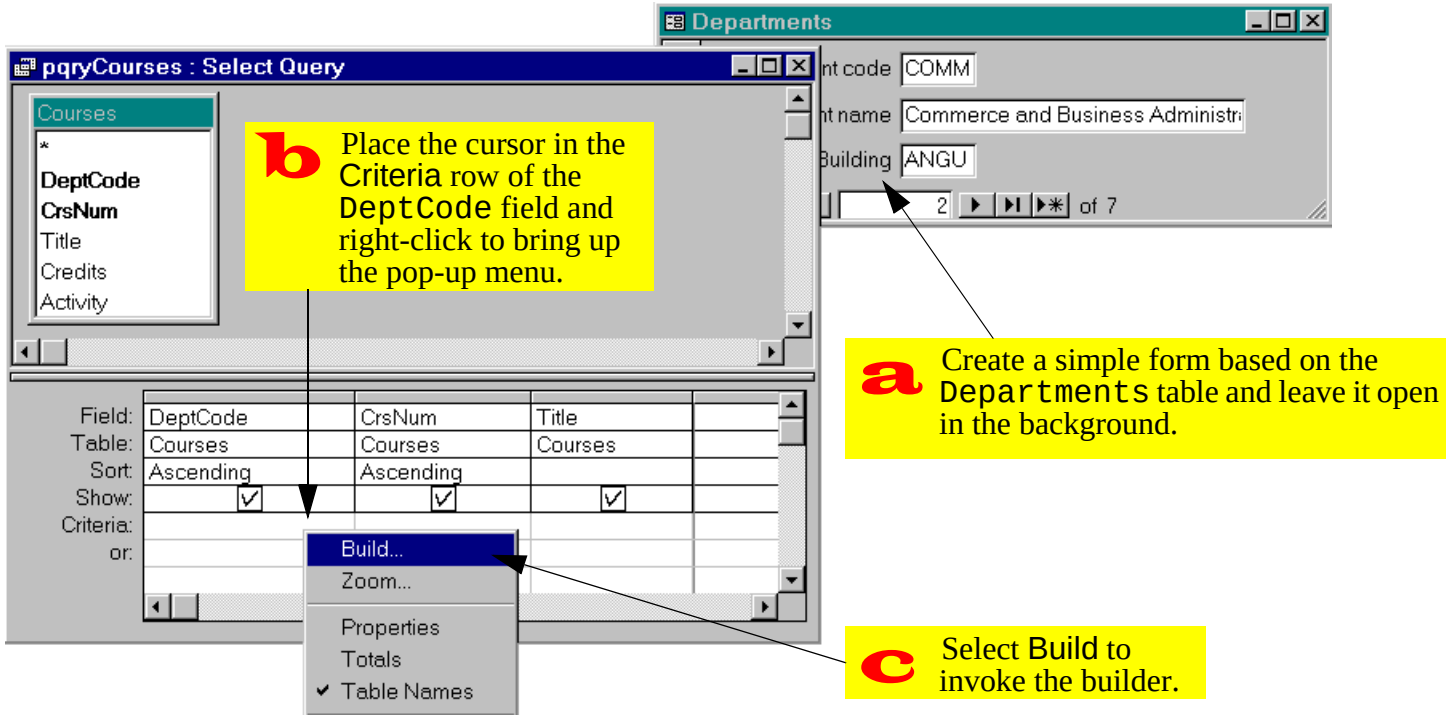


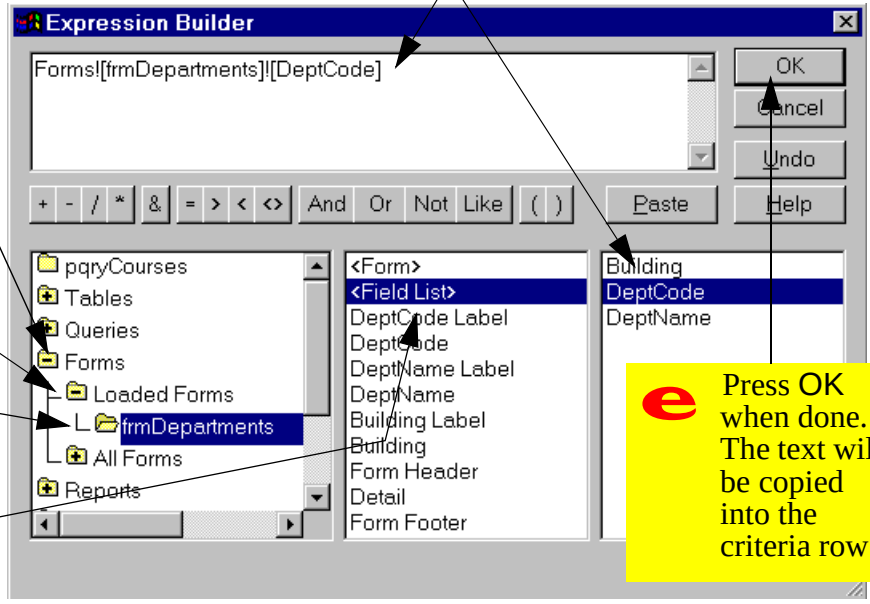
FIGURE 10.5: Use the builder to select the name of the object you want to use as a parameter.

a Select Forms to get a list of all the forms in your database.

b Since the `frmDepartments` form is open, click on Loaded Forms and select the form.

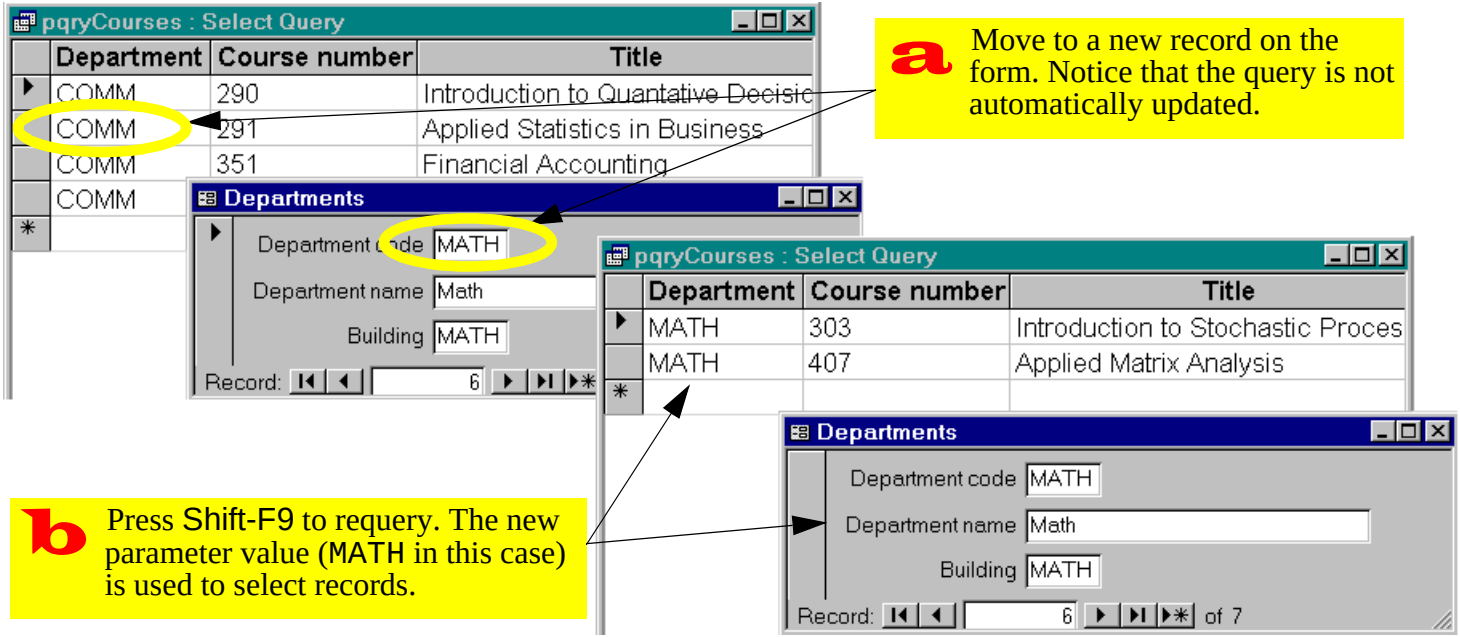
c Move to the middle pane and select Field List to get a list of the fields on the form in the pane on the far right.

d Double-click `DeptCode` to move it to the text area. If you make a mistake, move to the text area, delete the text, and try again.



e Press OK when done. The text will be copied into the criteria row.

FIGURE 10.6: Requery the results set to reflect changes on the form.



- Create a parameter query to show all the order details for a particular order.
- Create a second parameter query to show all the shipment details for a particular shipment.

Each order may result in a number of changes being made to the `BackOrders` table. For some items in the order, more product is ordered than is actually shipped (i.e., a backorder is created). For other items, more product is shipped than is ordered (i.e., a backorder is filled).

In Tutorial 15, you are supplied with a "shortcut" Visual Basic procedure that makes the changes to the `BackOrders` table for you. However, the shortcut procedure requires a query that lists the changes that must be made to the `BackOrders` table for a particular order. The requirements for this query are the following:

- The name of the query is `pqryItemsToBackOrder`
- It shows the change (positive or negative but not zero) in backorders for each item in a particular order.
- The query consist of three fields: `OrderID`, `ProductID` and a calculated field `Qty` (i.e., the change in the back order for a particular product).
- The name of the parameter is in this query is simply `[pOrderID]`. Since the value of this parameter will be set by the Visual Basic shortcut before the query is run, there is no need to set it to a value on a form.

Since the query is accessed by a program, the name of the query and all the fields must be *exactly as described above*. In other words, you are given a precise specification for a database object that fills a role in a process designed and implemented by someone else. You will not understand how the query fits in until Tutorial 15.